Supervised Learning on Logged Bandit Feedback

My Research Progress in Cornell

Reporter: Jinning Li
Adviser: Prof. Joachims

lijinning@sjtu.edu.cn; http://jinningli.cn
ACM Class, Shanghai Jiao Tong University

Logged Contextual Bandit Feedback

Logging Policy: π_0

Observed contex: $x_i \sim \Pr(X)$

Action: $y_i \sim \pi_0(Y \mid x_i)$

Logging Propensity: $p_i \equiv \pi_0(y_i \mid x_i)$

Feedback: $\delta_i \equiv \delta(x_i, y_i)$

$$D = [(x_1, y_1, p_1, \delta_1), \ldots, (x_n, y_n, p_n, \delta_n)]$$

Logged Contextual Bandit Feedback

Logging Policy: π_0

Observed contex: $x_i \sim \Pr(X)$

Action: $y_i \sim \pi_0(Y \mid x_i)$

Logging Propensity: $p_i \equiv \pi_0(y_i \mid x_i)$

Feedback: $\delta_i \equiv \delta(x_i, y_i)$

Objective: Train a network $\pi_w(Y \mid x)$

Minimizing

$$R(\pi_w) = \mathop{\mathbb{E}}_{x \sim \Pr(X)} \mathop{\mathbb{E}}_{y \sim \pi_w(Y|x)} [\delta(x, y)]$$

$$D = [(x_1, y_1, p_1, \delta_1), \ldots, (x_n, y_n, p_n, \delta_n)]$$

Example

Criteo Ad placement Dataset

Logging Policy: π_0 → Policy used by Criteo

Observed context: $x_i \sim \Pr(X)$ → Context of Ad candidates

Action: $y_i \sim \pi_0(Y \mid x_i)$ → Which candidate is chosen

Logging Propensity: $p_i \equiv \pi_0(y_i \mid x_i)$ → Propensity to choose it

Feedback: $\delta_i \equiv \delta(x_i, y_i)$ → Click: 1 else: 0

$$D = [(x_1, y_1, p_1, \delta_1), \ldots, (x_n, y_n, p_n, \delta_n)]$$

Example

Criteo Ad placement Dataset

896563753 |l 0.999 |p 336.294857951|f 0:300 1:600 2:1 3:1 4:1 5:1 6:1
896563753 |f 0:300 1:600 2:1 3:1 4:1 5:1 6:1 7:1 8:1 9:1 11:1
896563753 |f 0:300 1:600 2:1 12:1 13:1 14:1 21:1 27:1 34:1 35:1
896563753 |f 0:300 1:600 2:1 3:1 4:1 5:1 27:1 28:1 29:1 30:1 32:1

An impression

53638631 |l 0.999 |p 12.0914232512|f 0:2 1:2 2:1 6:1 11:1 12:1 13:1 14:1
53638631 |f 0:2 1:2 2:1 6:1 10:1 12:1 14:1 36:1 51:3 52:2 53:3 54:2 68:1 69:3 70:2 71:1
53638631 |f 0:2 1:2 2:1 6:1 12:1 52:3 53:2 54:3 69:2 70:3 77:1 78:1 79:1

Selected candidate(action)

746093459 |l 0.999 |p 11.3488158419|f 0:300 1:250 2:1 10:1 38:1 120:1 121:1 122:1
746093459 |f 0:300 1:250 2:1 67:1 101:1 106:1 109:1 124:1 128:1 129:1
746093459 |f 0:300 1:250 2:1 12:1 14:1 116:1 131:1 132:1 133:1 134:1 135:1

$$D = [(x_1, y_1, p_1, \delta_1), \ldots, (x_n, y_n, p_n, \delta_n)]$$

Method 1: Counterfactual Risk Minimization

Policy Optimizer for Exponential Models(POEM)

$$D = [(x_1, y_1, p_1, \delta_1), \ldots, (x_n, y_n, p_n, \delta_n)]$$

Objective:

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \delta_i \boxed{\min\{M, \frac{\boxed{\pi_w(y_i|x_i)}}{\pi_0}\}} + \boxed{\lambda \sqrt{\frac{Var_w(x)}{n}}}$$

$$\pi_w(y_i|x_i) = \frac{\exp(w \cdot \phi(x_i, y_i))}{\sum_{y_i' \in \mathcal{Y}_i} \exp(w \cdot \phi(x_i, y_i'))}$$

Clipping

Bias-Variance Tradeoff

Standard Deviation

Regularizer

vastly different variances of hypotheses

Counterfactual Risk Minimization: Learning from Logged Bandit Feedback; Adith Swaminathan, Thorsten Joachims; 2015

Method 2: Reduce to Supervised Learning (What I'm using)

$$D = [(x_1, y_1, p_1, \delta_1), \ldots, (x_n, y_n, p_n, \delta_n)]$$

1. Use supervised learning to estimate unseen feedback

```
896563753 |l 0.999 |p 336.294857951|f 0:300 1:600 2:1 3:1 4:1 5:1 6:1
896563753 |l ?.???  |f 0:300 1:600 2:1 3:1 4:1 5:1 6:1 7:1 8:1 9:1 11:1
896563753 |l ?.???  |f 0:300 1:600 2:1 12:1 13:1 14:1 21:1 27:1 34:1 35:1
896563753 |l ?.???  |f 0:300 1:600 2:1 3:1 4:1 5:1 27:1 28:1 29:1 30:1 32:1
```

Use the already seen feedback to train a learner to predict the unseen feedback

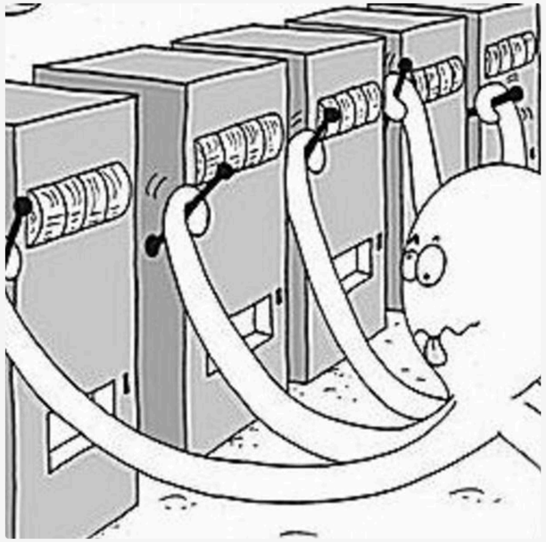
$$\arg\min_w \sum_{i=1}^{n} \mathcal{L}(\delta_i, h_w(x_i, y_i))$$ hypothesis $h \in \mathcal{H}$

2. Manually select a policy based on feedback,

For example, choose the candidate with **largest feedback**:

$$\hat{\pi} = \arg\max_{y_i \in \mathcal{Y}}(h(x_i, y_i))$$

Dataset & Evaluation Metric



NIPS '17 Workshop: Criteo Ad Placement Challenge

Counterfactual policy learning for display advertising

By Cornell University

Completed

9036 Views | 161 Participants | 242 Submissions

17 UNFOLLOW

Use this challenge as our dataset

Metric 1: IPS

$$IPS = \frac{10^4}{n^+ + 10n^-} \sum_{i=1}^{n} \delta_i \frac{\pi_w(\hat{y}_i|x_i)}{\pi_0} \qquad \pi_w(\hat{y}_i|x_i) = \frac{exp[h(x_i,\hat{y}_i) - \max h(x_i,y_i)]}{\sum_{y_i \in \mathcal{Y}_i} exp[h(x_i,y_i) - \max h(x_i,y_i)]}$$

Metric 2: IPS Standard Deviation

$$\frac{2.58 \times \sqrt{n}}{n^+ + 10n^-} \times Std[\delta \frac{\pi_w(\hat{y}_i|x_i)}{\pi_0}]$$

A Baseline: FOLLOW THE REGULARIZED LEADER (FTRL)

Used by Rank1 who won this challenge in 2017 ($2000 Prize!!)
Our model performs even better!!

An **online learning method**
Fast & Good for **sparse feature**

I reproduced his result, get
IPS=55.701; IPS_Std = 4.3

Find his **post-processing** is critical:

$$h(x, y) = \frac{850100}{1 + e^{-h(x,y)+1.1875}}$$

Algorithm 1 Per-Coordinate FTRL-Proximal with L_1 and L_2 Regularization for Logistic Regression

 # With per-coordinate learning rates of Eq. (2).
Input: parameters α, β, λ_1, λ_2
$(\forall i \in \{1, \ldots, d\})$, initialize $z_i = 0$ and $n_i = 0$
for $t = 1$ **to** T **do**
  Receive feature vector $\mathbf{x}_t$ and let $I = \{i \mid x_i \neq 0\}$
  For $i \in I$ compute

$$w_{t,i} = \begin{cases} 0 & \text{if } |z_i| \leq \lambda_1 \\ -\left(\frac{\beta+\sqrt{n_i}}{\alpha} + \lambda_2\right)^{-1}(z_i - \text{sgn}(z_i)\lambda_1) & \text{otherwise.} \end{cases}$$

  Predict $p_t = \sigma(\mathbf{x}_t \cdot \mathbf{w})$ using the $w_{t,i}$ computed above
  Observe label $y_t \in \{0, 1\}$
  for all $i \in I$ **do**
    $g_i = (p_t - y_t)x_i$ *#gradient of loss w.r.t.* w_i
    $\sigma_i = \frac{1}{\alpha}\left(\sqrt{n_i + g_i^2} - \sqrt{n_i}\right)$ *#equals* $\frac{1}{\eta_{t,i}} - \frac{1}{\eta_{t-1,i}}$
    $z_i \leftarrow z_i + g_i - \sigma_i w_{t,i}$
    $n_i \leftarrow n_i + g_i^2$
  end for
end for

https://github.com/alexeygrigorev/nips-ad-placement-challenge
Ad Click Prediction: a View from the Trenches, H. Brendan McMahan et. al. 2013

Why his post-processing is critical?

$$h(x,y) = \frac{850100}{1 + e^{-h(x,y)+1.1875}}$$

Most participants treat this challenge as a **CTR prediction challenge**

Their output $\tilde{\delta} \in [0,1]$

However, the **evaluation program** will apply Softmax to get π_w

$$\pi_w(\hat{y}_i|x_i) = \frac{exp[h(x_i,\hat{y}_i) - \max h(x_i,y_i)]}{\sum_{y_i \in \mathcal{Y}_i} exp[h(x_i,y_i) - \max h(x_i,y_i)]}$$

Without post-processing, their π_w in evaluation program look like:

[0.984536, 0.980575, 0.984368, 0.974204, 0.97577, 0.986368, 0.976827, 1, 0.984053, 0.995168, 1]

Using post-processing, their π_w in evaluation program look like:

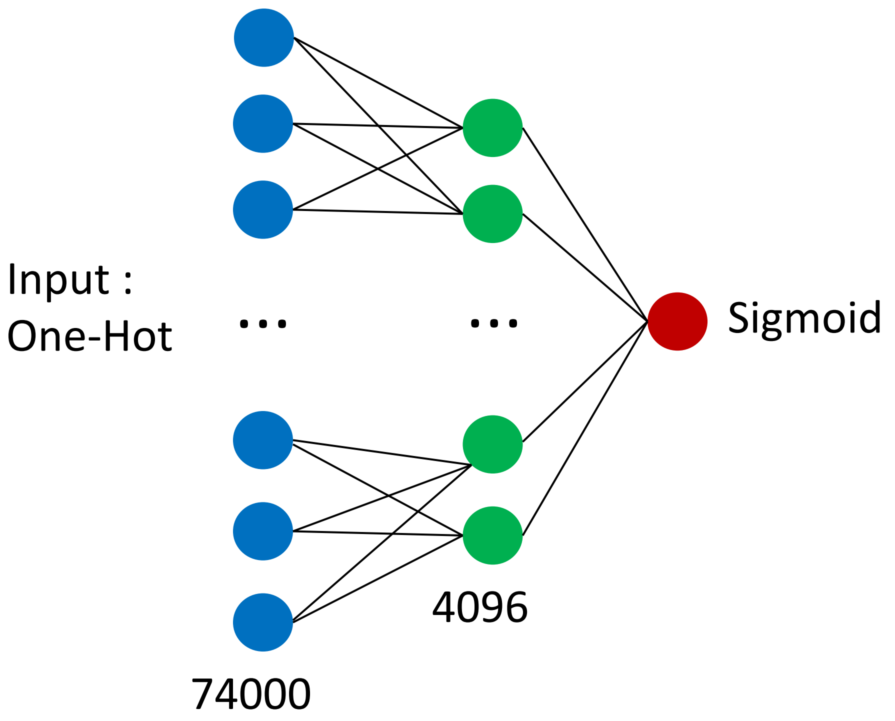
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 3.05902e-07]

Higher variance,

but higher IPS score, and more reasonable

I also apply this post-processing in my model

My method: Logistic Regression (MLP): Version 1.0



Loss Function:

$$\mathcal{L}_h = -\left[\delta \cdot \log h(x, y) + (1 - \delta) \cdot \log(1 - h(x, y))\right]$$

Policy: equivalent to

$$\hat{\pi} = \arg\max_{y_i \in \mathcal{Y}} (h(x_i, y_i))$$

Because after post-processing,

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 3.05902e-07]

Result: **IPS=59.249 IPS_Std=5.46** (2 Epoch)

Better than FTRL

Reasonable --- FTRL is online learning

https://github.com/jinningli/ad-placement-pytorch

My method: LR & Propensity: Version 2.0

New Loss Function:

$$\mathcal{L}_h = -\frac{1}{p_0}\left[\delta \cdot \log h(x,y) + (1-\delta)\cdot \log(1-h(x,y))\right]$$

Intuition:

We only observe the feedback of candidate selected by π_0

However, if π_0 has high propensity for some candidates, they will always be selected. This will cause bias.

So, we introduce this **inverse propensity weighting** to reduce bias.

Result: **IPS=54.546 IPS_Std=2.943**

Problem 1:

Higher variance

Problem 2:

p_0 estimated by π_0 is related to x_i

i.e. other candidates in i-th impression

https://github.com/jinningli/ad-placement-pytorch

My method: LR & Propensity & clipping: Version 3.0

Loss Function **with clipping**:

$$\mathcal{L}_h = -\min\{M, \frac{1}{p_0}\}\,[\delta \cdot \log h(x,y) + (1-\delta)\cdot \log(1-h(x,y))]$$

Intuition:

Control the gradient when $\frac{1}{p_0}$ goes too large.

Control the bias-variance tradeoff.

Actually $\frac{1}{p_0}$ can be **quite large**, varying between [1,2917566], Average: 114.57

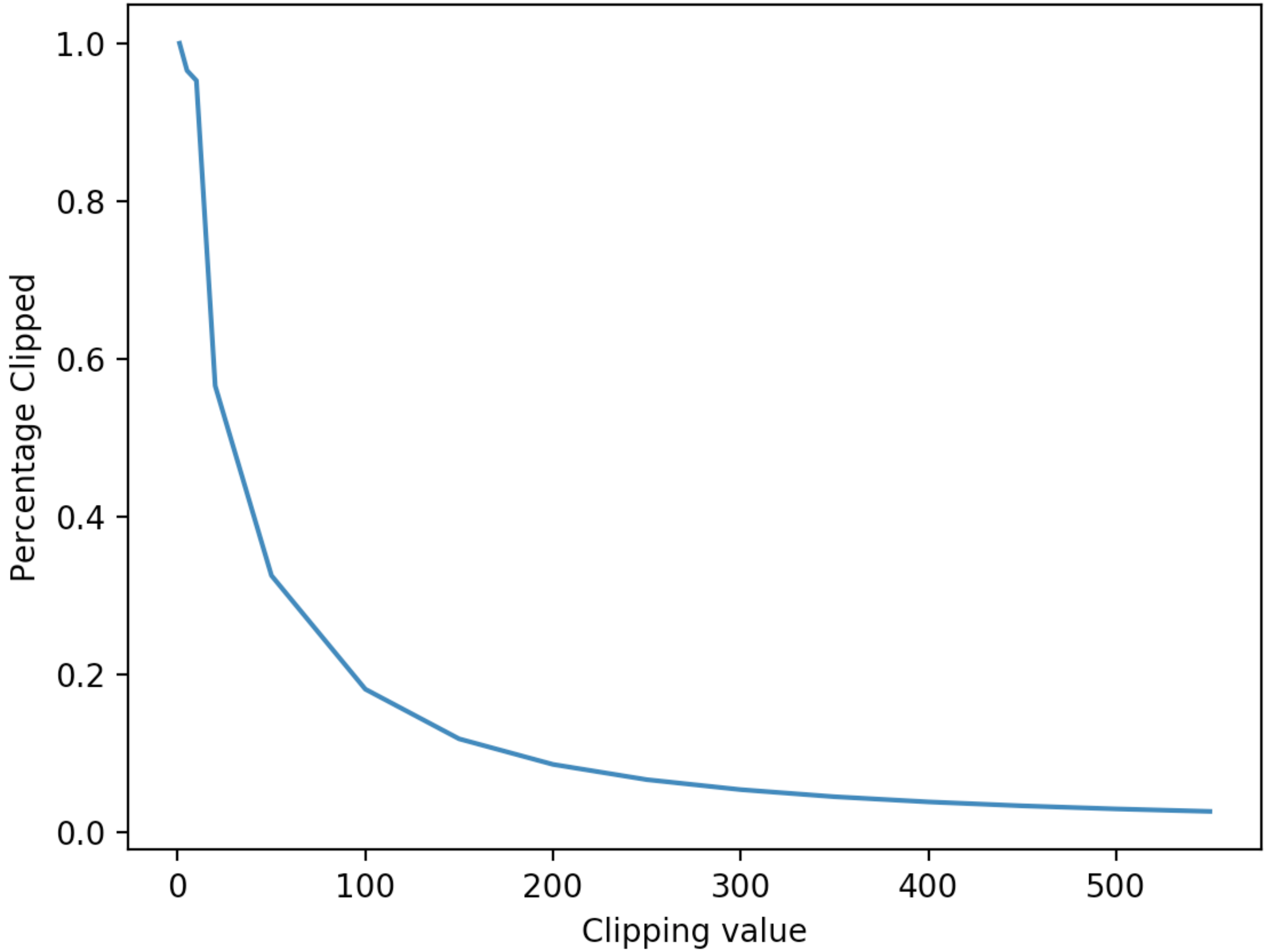
Problem 1:
Higher variance

* Maybe we should use $\min\{M, \frac{114.57}{p_0}\}$?

https://github.com/jinningli/ad-placement-pytorch

My method: LR & Propensity & clipping: Version 3.0

How many propensity is clipped with threshold M?



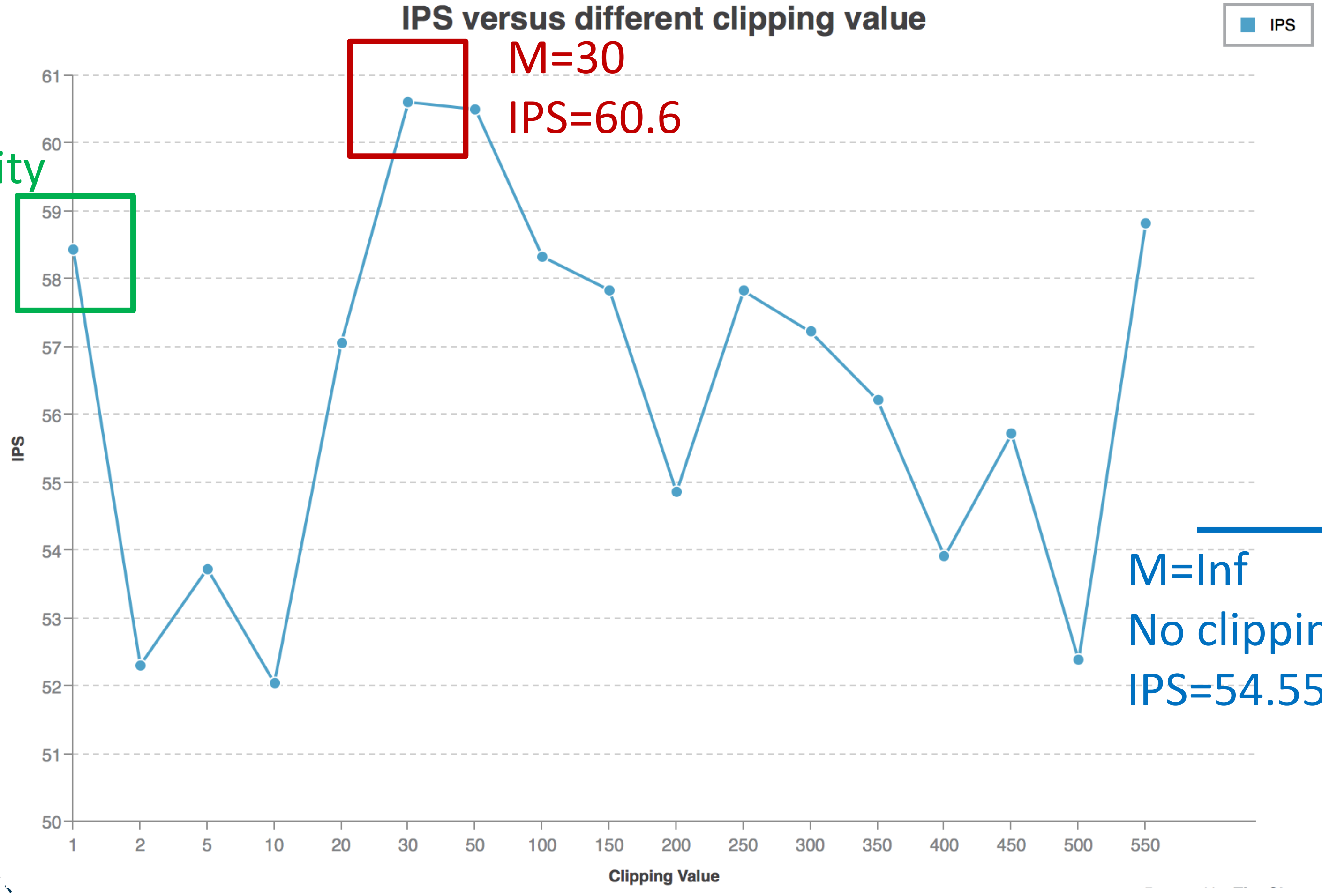
So I take
M= [1, 2, 5, 10, 15, 20, 30, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550]
to see how the IPS score varies.

https://github.com/jinningli/ad-placement-pytorch

My method: LR & Propensity & clipping: Version 3.0

Performance of clipping?

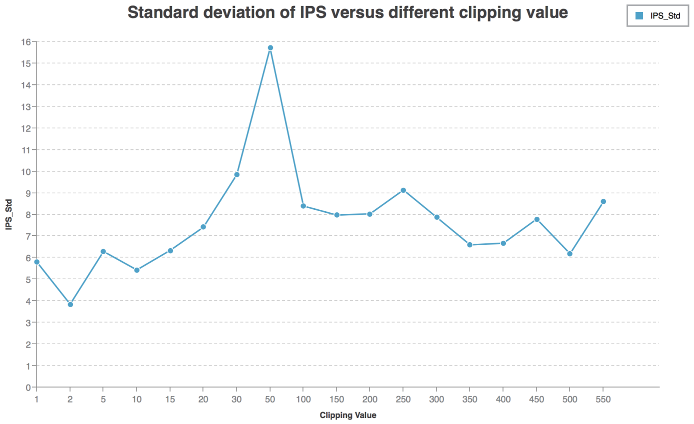
No Propensity
(All clipped)
IPS=58.4



M=Inf
No clipping
IPS=54.55

When **30<M<50**, the IPS is the highest (over 60). However, the standard deviation is also quite high (over 10).

Anyway, clipping is effective to reduce bias



https://github.com/jinningli/ad-placement-pytorch

My method: LR & now propensity : Version 4.0

Loss Function with now propensity:

$$\mathcal{L}_h = -\frac{p_w}{p_0}\left[\delta \cdot \log h(x,y) + (1-\delta)\cdot \log(1-h(x,y))\right]$$

Here p_w is not embedded inside the computational graph (**no gradient**)

Input all the candidates in an impression to calculate p_w

$$p_{\tilde{w}}(y_i|x_i) = \frac{exp[\tilde{h}(x_i,\hat{y}_i) - \max \tilde{h}(x_i,y_i)]}{\sum_{y_i \in \mathcal{Y}_i} exp[\tilde{h}(x_i,y_i) - \max \tilde{h}(x_i,y_i)]}$$

Intuition: Reduce bias and also control variance.

Result: Not good. But variance is controlled **IPS = 52.95 IPS_Std = 2.42**

Problem 2:

p_0 estimated by π_0 is related to x_i

i.e. other candidates in i-th impression

https://github.com/jinningli/ad-placement-pytorch

What if we embed p_w to the graph?   Version 5.0

Loss Function is still:

$$\mathcal{L}_h = -\frac{p_w}{p_0}\boxed{[\delta \cdot \log h(x,y) + (1-\delta)\cdot \log(1-h(x,y))]}$$

The performance is improved much: **Result: IPS=61.69 IPS_Std=27.06**

Actually, now in Version 5, our objective is quite similar to POEM.

Recall the objective of PORM:

$$\arg\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \boxed{\delta_i} \min\{M, \frac{\pi_w(y_i|x_i)}{\pi_0}\} + \lambda\sqrt{\frac{Var_w(x)}{n}}$$

The only difference is I **minimize the Cross Entropy of feedback** while POEM uses the **logged feedback as a constant**.

https://github.com/jinningli/ad-placement-pytorch

I’m still not satisfied! The variance is still high! Version 6.0

What if we add a regularizer to loss function?

$$\mathcal{L}_h = -\frac{p_w}{p_0}\left[\delta \cdot \log h(x,y) + (1-\delta)\cdot \log(1-h(x,y))\right] + \left(tanh^2\left(\frac{1}{p_w}\right) - tanh^2\left(\frac{1}{p_0}\right)\right)^{\frac{1}{2}}$$

Result: not good. **IPS=52.46 IPS_Std=5.41**

$\times\lambda$

Maybe:

1. MSE+tanh function is not suitable
2. The regularizer should be weighted
3. I should try the same regularizer with POEM:

$$\lambda\sqrt{\frac{Var_w(x)}{n}}$$

https://github.com/jinningli/ad-placement-pytorch

Conclusions

Model	IPS	IPS Standard Deviation
FTRL	55.70	4.30
Logistic Regression(LR)	59.25	5.46
LR & propensity	54.55	2.94
LR & propensity clipping	60.60	15.80
LR & now propensity (no gradient)	52.95	**2.42**
LR & now propensity (with gradient)	**61.69**	27.06
LR & now propensity & regularizer	52.46	5.41

$$\mathcal{L}_h = -\frac{p_w}{p_0}\left[\delta \cdot \log h(x,y) + (1-\delta)\cdot \log(1-h(x,y))\right]$$

Future work?

Compare my method with other method such as POEM.

Build **another reliable dataset** for experiments; Find other interesting task where counterfactual learning can be applied.

Investigate the **relationship** between **clipping threshold** M and distribution of logged inverse propensity.

Other **network architecture** (convolutional networks)?

Thank You! 

Questions are welcome

Reporter: Jinning Li

lijinning@sjtu.edu.cn; http://jinningli.cn

ACM Class, Shanghai Jiao Tong University